IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Gerard CHAUVEL, et al.	§	Confirmation No.:	9347
Serial No.:	10/632,024	§	Group Art Unit:	2109
Filed:	July 31, 2003	§	Examiner:	J. R. Swearingen
For:	Synchronization of Processor States	§	TI Docket No.:	TI-35461

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Date: November 13, 2007

Sir:

Appellants hereby submit this Appeal Brief in connection with the above-identified application. A Notice of Appeal is filed concurrently herewith.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is Texas Instruments Inc., a Delaware corporation, having its principal place of business in Dallas, Texas. The Assignment from the inventors to Texas Instruments-France was recorded on July 31, 2003, at Reel/Frame 014355/0278. The assignment from Texas Instruments-France to Texas Instruments Inc. was recorded on March 12, 2004 at Reel/Frame 014421/0927.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

Originally filed claims:	1-25.
Claim cancellations:	None.
Added claims:	None.
Presently pending claims:	1-25.
Presently appealed claims:	1-25.

IV. STATUS OF THE AMENDMENTS

No claims were amended after the final Office action dated August 22, 2007.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The specification is directed to synchronization of processor states.[1] At least some of the illustrative embodiments are systems as in claim 1:

1. A system, comprising:
   a first processor that executes a transaction targeting a pre-determined address;[2]
   a second processor coupled to said first processor;[3] and
   a wait unit coupled to said first and second processors, said wait unit detects said pre-determined address and asserts a wait signal to cause said first processor to enter a wait mode.[4]

Other embodiments are methods as in claim 9:

9. A method, comprising:
   executing a transaction that targets a pre-determined address;[5]
   detecting the transaction to said pre-determined address;[6]
   asserting a wait signal upon detection of the transaction to cause a processor to stall;[7]
   causing said wait signal to de-assert upon occurrence of an event, said de-assert controlled by logic external to said processor.[8]

Yet still other embodiments are wait units as in claim 15:

15. A wait unit, comprising:
   a decode logic unit that determines when a first processor runs a transaction to a pre-determined address;[9]
   a first processor interface;[10]

---

[1] Specification Title.

[2] Specification Page 7, paragraph [0018], lines 1-3 within the paragraph. A shorthand notation for citations to the specification takes the form ([page], [paragraph], lines [lines within the paragraph]). The citation of this footnote in the shorthand forms reads: (7, [0028], lines 1-3). *See also*, (7, [0019], lines 1-12), Figure 2a, element 202.

[3] (6, [0017], lines 1-5), Figure 2a, element 205

[4] (7, [0019], lines 9-12), Figure 2a, element 206.

[5] (12, [0034], lines 2-3), Figure 4, element 302.

[6] (12, [0034], lines 3-4), Figure 4, element 304.

[7] (12, [0034], lines 3-4), Figure 4, element 306.

[8] (12, [0034], lines 4-6), Figure 4, elements 308 and 310.

[9] (11, [0030], lines 1-2), Figure 3, element 220.

a second processor interface;[11]
logic[12] coupled to the decode logic unit, the first processor interface, and the second processor interface, wherein said logic asserts a signal propagated by the first processor interface to cause said first processor to stall.[13]

Other embodiments are systems as in claim 21:

21. A system, comprising:
a first processor;[14]
a second processor;[15]
means for detecting[16] a transaction targeting a pre-determined address[17] and for asserting a wait signal to said first processor to cause the first processor to enter a wait state;[18] and
means for releasing[19] said first processor from the wait state.[20]

---

[10] (11, [0029], lines 1-3), Figure 3, element 222.

[11] (11, [0029], lines 1-3), Figure 3, element 224.

[12] (11, [0029], lines 1-3), Figure 3, element 228.

[13] (11, [0030], lines 5-8).

[14] (6, [0017], lines 2-5), Figure 2a, element 202.

[15] (6, [0017], lines 2-5), Figure 2a, element 204.

[16] This limitation is specifically identified as a means-plus-function limitation under 35 USC § 112, sixth paragraph.

[17] (11, [0030], lines 1-2), Figure 3, element 220.

[18] (11, [0030], lines 5-8), Figure 3, element 228

[19] This limitation is specifically identified as a means-plus-function limitation under 35 USC § 112, sixth paragraph.

[20] (11, [0032], lines 3-10), Figure 3, element 228.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1-3, 7, 9-10 12, 21 and 24 are anticipated under 35 USC §102(e) over Evoy et al. (U.S. Pat. No. 6,766,460, hereafter Evoy).

Whether claims 15 and 18 are anticipated under 35 USC §102(b) over Miller et al. (U.S. Pat. No. 6,766,460, hereafter Miller).

Whether claims 4-5, 13-14 and 22-23 are obvious under 35 USC §103(a) over Evoy and Shenk (U.S. Pat. No. 4,535,404).

Whether claims 6, 11 and 25 are obvious under 35 USC §103(a) over Evoy and Mustafa (U.S. Pat. No. 6,678,830).

Whether claim 8 is obvious under 35 USC §103(a) over Evoy and Johnson Jr. et al. (U.S. Pat. No. 4,420,806, hereafter Johnson).

Whether claims 19 is obvious under 35 USC §103(a) over Miller and Johnson.

VII. ARGUMENT

A. Section 102 Rejections Over Evoy

1. Claims 1-3, 7, 9-10 12, 21 and 24

Claims 1-3, 7, 9-10 12, 21 and 24 stand rejected as allegedly anticipated by Evoy. Claim 1 is representative of this grouping of claims. The grouping should not be construed to mean the patentability of any of the claims may be determined in later actions (*e.g.*, actions before a court) based on the groupings. Rather, the presumption of 35 USC § 282 shall apply to each of these claims individually.

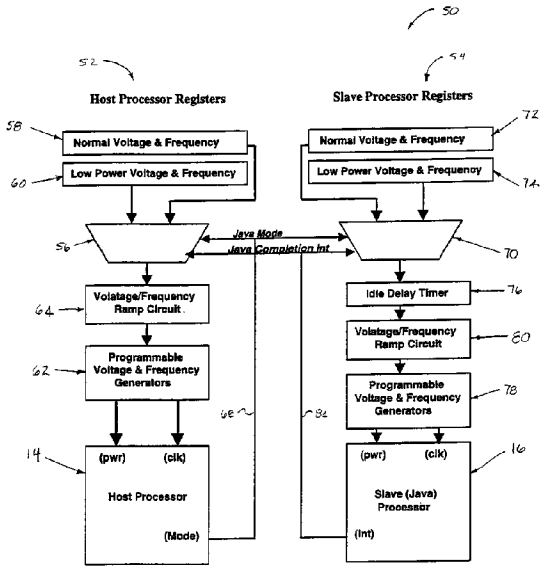
Evoy is directed to a system and method for power management in a Java accelerator environment.[21] In particular, Evoy discloses having a master or host processor 14 and a slave or Java processor 16.[22] The Evoy system is configured to (among other things): power the host processor 14 and to reduce power to the Java processor 16 when non-Java code is executed: and to power the Java processor 16 and to reduce power the host processor 14 when Java code is executed.[23] In order to discuss the process of switching between executing on host processor 14 and the Java processor 16, Evoy's Figure 2 is reproduced immediately below.

---

[21] Evoy Title.

[22] Evoy Col. 4, lines 14-21; Figure 1.

[23] Evoy Col. 4, lines 1-5 ([The Evoy system] manages power usage in a Java accelerator by reducing power to a host processor and increasing power to a Java processor when a Java application is initiated, and reversing the process when execution of the Java application is halted.")



When Evoy's host processor 14 encounters Java code, the host processor 14 asserts Java Mode signal 68 (vertical line, middle of the figure) which switches the power mode on both the host and java processors.[24] In particular,

> Upon receiving the Java mode signal 68, the host register switch 56 sets the host target voltage and frequency values for the host processor 14 to the values stored in the host low power and frequency registers 60. Similarly, the Java register switch 70 sets the Java target voltage and frequency values for the Java processor 16 to the values stored in the Java normal operating voltage and frequency registers 72.[25]

Thus, it appears that Evoy's host processor 14 makes the determination as to whether to switch execution modes based on encountering Java code, and the decision is evidenced by the assertion of a Boolean value – the Java Mode signal 68.

Representative claim 1 recites, "a first processor that executes a transaction targeting a pre-determined address; … and a wait unit coupled to said first and second processors, said wait unit detects said pre-determined address and asserts a wait signal to cause said first processor to enter a wait mode."

---

[24] Evoy Col. 6, lines 7-22.

[25] Evoy Col. 6, lines 23-32.

Appellants respectfully submit that Evoy fails to expressly or inherently teach such a system. In particular, Evoy fails to expressly or inherently teach a wait unit with the defined characteristics, as in Evoy it is the host processor 14 that determines that a switch to Java mode is needed (thus reducing power to the host processor). The Office action of August 22, 2007 attempts to rely on Evoy Col. 6, lines 15-35 for this limitation, but the reliance is misplaced. As best understood, the Office action relies on the power management system 50 (Figure 2) of Evoy for the claimed wait unit; however, it is the host processor 14 that asserts the Boolean Java Mode signal 68 to cause the switch. Thus, Evoy fails to expressly or inherently teach a "wait unit **coupled to said first and second processors, said wait unit detects said pre-determined address and asserts a wait signal** to cause said first processor to enter a wait mode." For this reason alone the rejection should be withdrawn.

Moreover, claim 1 recites "a first processor that executes a transaction targeting a pre-determined address." In Evoy, the switch between host processor 14 and the Java processor 16 executing is based on the reading of a Java method or Java bytecodes independent of address. Evoy is silent as to predetermined addresses for the Java code. As the Board is no doubt aware, a Java method or bytecodes may reside in virtually any location in the memory space, and thus Evoy fails to expressly or inherently teach that the transition to a different power mode should be based on the first processor making "a transaction to a **pre-determined address**."

Based on the foregoing, Appellants respectfully request that the rejection of this first grouping be reversed, and the claims set for issue.

B. Section 102 Rejections Over Miller

1. Claims 15 and 18

Claims 15 and 18 stand rejected as allegedly anticipated by Miller. Claim 15 is representative of this grouping of claims. The grouping should not be construed to mean the patentability of any of the claims may be determined in later actions (*e.g.*, actions before a court) based on the groupings. Rather, the presumption of 35 USC § 282 shall apply to each of these claims individually.

Miller is directed to a method for initializing a multiple processor computer system using a common ROM.[26] In particular, Miller teaches:

> Two design variations which allow multiple processor to start up using a ROM… . In each design, a single, primary processor is allowed to perform a complete POST while the remaining, secondary processors are directed in the course of their POST to perform a more limited initialization sequence. At power on, the primary processor begins a normal POST, while the secondary processors are held until a vector is placed into a redirection vector location. Each secondary processor is then subsequently started, using its own initialization code located at the address indicated by the redirection vector.[27]

In addition to performing POST for each processor as discussed in the quotation above, Miller discloses methods of allocating various tasks to the processors once those processors have completed POST.[28]

Representative claim 15 recites a wait unit comprising "a decode logic unit that determines when a first processor runs a transaction to a pre-determined address… ." The Office action relies on Miller's interprocessor logic 30 for the claimed decode logic, citing Col. 5, line 57 – Col. 6, line 6, reproduced immediately below.

> The second design utilizes a sleep bit located in the Processor Option Register in the interprocessor logic 30 associated with each processor P_Z. The sleep bit operates such that, when it is set for a respective processor, requests for the bus 40 by the respective processor are blocked. Therefore, in the second design, the processors P_1 and P_Z each have their reset bit toggled and a sleep bit is set on the processor P_Z. The toggling of the reset bit of the processor P_1 allows it to begin the POST, while the setting of the sleep bit on the processor P_Z, which occurs at power up by hardware control, causes any requests for the bus by the processor P_Z to be blocked, thus effectively placing the processor P_Z in a held state. **A subsequent clearing of the sleep bit of the processor P_Z by the processor P_1** allows bus requests by the processor P_Z to be passed, thereby allowing the processor P_Z to begin the POST.[29]

---

[26] Miller Title.

[27] Miller Abstract.

[28] Miller Col. 9, lines 54-65.

[29] Miller Col. 5, line 57 – Col. 6, line 6 (emphasis added).

Thus, with respect to allowing processor P_Z to perform POST operations, Miller expressly teaches that processor P_1 clears the sleep bit for processor P_Z in the interprocessor logic. Thus, Miller fails to expressly or inherently teach a wait unit comprising "a decode logic unit that determines when a first processor runs a transaction to a pre-determined address… ." For this reason alone the rejections should be withdrawn and the claims set for issue.

Moreover, representative claim 15 recites, "logic coupled to the decode logic unit …, wherein said logic asserts a signal … to cause said first processor to stall." The Office action relies on Miller's Col. 10, directed to allocation of tasks to processors that have already completed POST, the cited location reproduced immediately below.

> If the availability bit of the processor P_2 is not set, then the operating system knows that either the processor P_2 has not yet finished the task that it was previously given or that the processor P_2 is not presently configured to the system. In either case, the processor P_2 is determined to be unavailable. If the availability bit of the processor P_2 is set, then the operating system commences with the allocation of the task in step 284 by placing a vector in memory location 40:67 pointing to software which generally includes the task that the operating system wishes the processor P_2 to execute. The processor P_1 saves the previous value from memory location 40:67 for later restoration.
>
> **The processor P_1 then activates the processor P_2 in step 286 by clearing the sleep bit in the processor P_2's Processor Option Register located in the interprocessor logic 30** in step 286, causing the processor P_2 in step 290 to obtain the vector at memory location 40:67 to begin operation of the new task. An attempt to obtain the vector at memory location 40:67 was actually the last instruction executed by the processor P_2 in the POST procedure in step 270. This vector fetch is by definition a cache miss so that the processor P_2 must use the bus. However, the sleep bit was set in step 270, disabling any bus request, so the vector fetch is pending, waiting for the sleep bit to be cleared. Once the sleep bit is cleared, the processor P_2 can properly access the bus and obtain the vector at memory location 40:67 pointing to the task that it is to execute. The first instruction in this new task is step 292 which directs the processor P_2 to clear its respective availability bit in the available bit register to indicate to the operating system that it is no longer available for dispatching and that it has commenced with the

> task. After this, the processor P_2 begins executing the task in step 294.
>
> The clearing of the availability bit by the processor P_2 serves as notification to the processor P_1 that the processor P_2 has begun execution of the task that the oprating [sic] system has given it. Upon receiving this notification, in step 296 the processor P_1 restores the previous vector to the reset vector memory location 40:67 and then resumes the execution of its own code. **When the processor P_2 has finished executing its task in step 294, *it* sets the sleep bit in its Processor Option Register** in step 298… .[30]

In Miller, processor P_1 clears the sleep bit, and processor P_2 sets the sleep bit. Thus, no portion of Miller's interprocessor logic 30 could be considered a "logic coupled to the decode logic unit …, wherein **said logic asserts a signal … to cause said first processor to stall.**"

Based on the foregoing, Appellants respectfully request that the rejection of this first grouping be reversed, and the claims set for issue.

C. Section 103 Rejections Over Evoy and Shenk

1. Claims 4-5, 13-14 and 22-23

Claims 4-5, 13-14 and 22-23 are patentable for at least the same reasons as discussed in Section VII(A)(1).

D. Section 103 Rejection Over Evoy and Mustafa

1. Claims 6, 11 and 25

Claims 4-5, 13-14 and 22-23 are patentable for at least the same reasons as discussed in Section VII(A)(1).

E. Section 103 Rejections Over Evoy and Johnson

1. Claim 8

Claims 4-5, 13-14 and 22-23 are patentable for at least the same reasons as discussed in Section VII(A)(1).

F. Section 103 Rejections Over Miller and Johnson

1. Claim 19

Claim19 is patentable for at least the same reasons as discussed in Section VII(B)(1).

---

[30] Miller Col. 10, lines 12-47 (emphasis added).

G. Conclusion

For the reasons stated above, Appellants respectfully submit that the Examiner erred in rejecting all pending claims. It is believed that no extensions of time or fees are required, beyond those that may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, such extensions are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required (including fees for net addition of claims) are hereby authorized to be charged to the Texas Instruments, Inc. Deposit Account No. 20-0668.

Respectfully submitted,

/Mark E. Scott/

Mark E. Scott
PTO Reg. No. 43,100
CONLEY ROSE, P.C.
(512) 391-1900 (Phone)
(512) 320-9181 (Fax)
ATTORNEY FOR APPELLANTS

VIII. CLAIMS APPENDIX

1. (Original) A system, comprising:

a first processor that executes a transaction targeting a pre-determined address;

a second processor coupled to said first processor; and

a wait unit coupled to said first and second processors, said wait unit detects said pre-determined address and asserts a wait signal to cause said first processor to enter a wait mode.

2. (Original) The system of claim 1 wherein the wait signal is de-asserted to permit the first processor to retrieve a status of the second processor.

3. (Original) The system of claim 2 wherein said status includes one or more instructions that the first processor is to execute.

4. (Original) The system of claim 1 wherein said transaction comprises a read instruction.

5. (Original) The system of claim 1 wherein said transaction comprises a write instruction.

6. (Original) The system of claim 1 wherein said wait unit de-asserts the wait signal upon detection of a system interrupt signal generated by the first processor.

7. (Original) The system of claim 1 wherein said wait unit de-asserts the wait signal upon detection of a signal from said second processor.

8. (Original) The system of claim 7 wherein said wait unit upon detection of said signal asserts a processor interrupt signal to the first processor if the wait signal is already de-asserted.

9. (Original) A method, comprising:

executing a transaction that targets a pre-determined address;

detecting the transaction to said pre-determined address;

asserting a wait signal upon detection of the transaction to cause a processor to stall;

causing said wait signal to de-assert upon occurrence of an event, said de-assert controlled by logic external to said processor.

10. (Original) The method of claim 9 wherein said stall comprises a low power mode.

11. (Original) The method of claim 9 wherein said event comprises a system interrupt.

12. (Original) The method of claim 9 wherein said event comprises a signal from another processor.

13. (Original) The method of claim 9 wherein said transaction is a read instruction to said pre-determined address.

14. (Original) The method of claim 9 wherein said transaction is a write instruction to said pre-determined address.

15. (Previously Presented) A wait unit, comprising:

a decode logic unit that determines when a first processor runs a transaction to a pre-determined address;

a first processor interface;

a second processor interface;

logic coupled to the decode logic unit, the first processor interface, and the second processor interface, wherein said logic asserts a signal propagated by the first processor interface to cause said first processor to stall.

16. (Original) The wait unit of claim 15 wherein said transaction is a read instruction.

17. (Original) The wait unit of claim 15 wherein said transaction is a write instruction.

18. (Original) The wait unit of claim 15 wherein said second processor interface receives a wait release signal from a second processor that causes the wait unit to de-assert the wait signal to said first processor through said first processor interface.

19. (Original) The wait unit of claim 18 wherein said wait release signal causes a processor interrupt signal to be asserted if the wait signal is already de-asserted.

20. (Original) The wait unit of claim 15 further comprising a system interrupt interface coupled to the logic, through which a system interrupt signal is received that causes the logic to de-assert said wait signal to said first processor through the first processor interface.

21. (Original) A system, comprising:

a first processor;

a second processor;

means for detecting a transaction targeting a pre-determined address and for asserting a wait signal to said first processor to cause the first processor to enter a wait state; and

means for releasing said first processor from the wait state.

22. (Original) The system of claim 21 wherein the transaction comprises a memory read.

23. (Original) The system of claim 21 wherein the transaction comprises a memory write.

24. (Original) The system of claim 21 wherein said means for releasing said first processor from the wait state comprises a wait release signal from said second processor coupled to a wait unit, said wait unit de-asserts the wait signal upon detection of the wait release signal.

25. (Original) The system of claim 21 wherein said means for releasing said first processor from the wait state comprises a system interrupt signal to a wait unit, said wait unit de-asserts the wait signal upon detection of the system interrupt signal.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.